**深 圳 大 学 实 验 报 告**

**课 程 名 称： 计算机系统(3)**

**实验项目名称： 五级流水线与流水线冒险**

**学 院： 计算机与软件学院**

**专 业： 计算机科学与技术**

**指 导 教 师： 周明洋**

**报告人： 杜良衡 学号： 2022150255 班级： 01B**

**实 验 时 间： 2024.11.8**

**实验报告提交时间： 2024.11.8**

**教务处制**

**一、 实验目标：**

了解MIPS的五级流水线，和在运行过程中的所产生的各种不同的流水线冒险

通过指令顺序调整，或旁路与预测技术来提高流水线效率

更加了解流水线细节和其指令的改善方法

更加熟悉MIPS指令的使用

**二、实验内容**

观察一段代码并运行，观察其中的流水线冒险，并记录统计信息

对所给的代码进行指令序列的调整，以期避免数据相关，并记录统计信息

启动forward功能，以获得性能提升，并且记录统计信息

**三、实验环境**

硬件：桌面PC

软件：Windows、WinMIPS64

**四、****实验步骤及说明**

首先，给出一段C代码，该段代码实现的是两个矩阵相加。

设有4\*4矩阵A和4\*4矩阵B相加，得到4\*4矩阵C：

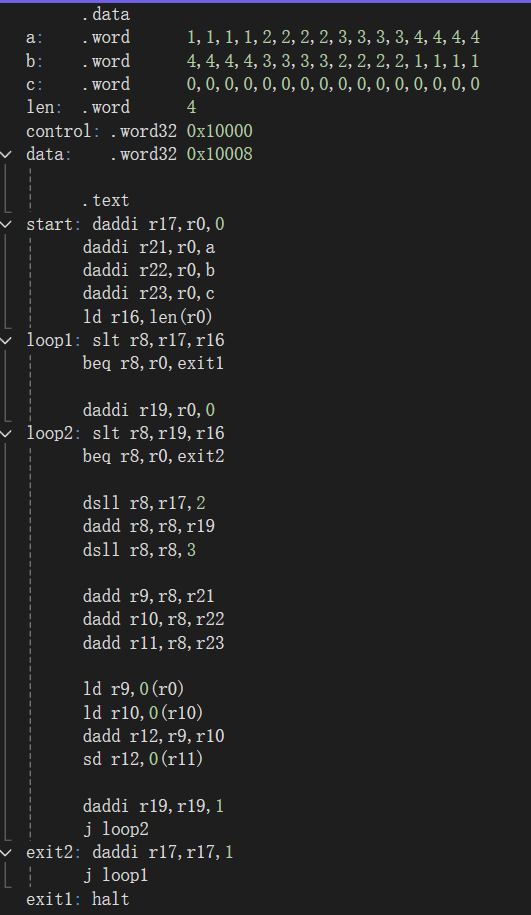
for(int i = 0; i < 4; i++)

For(int j = 0; j < 4; j++)

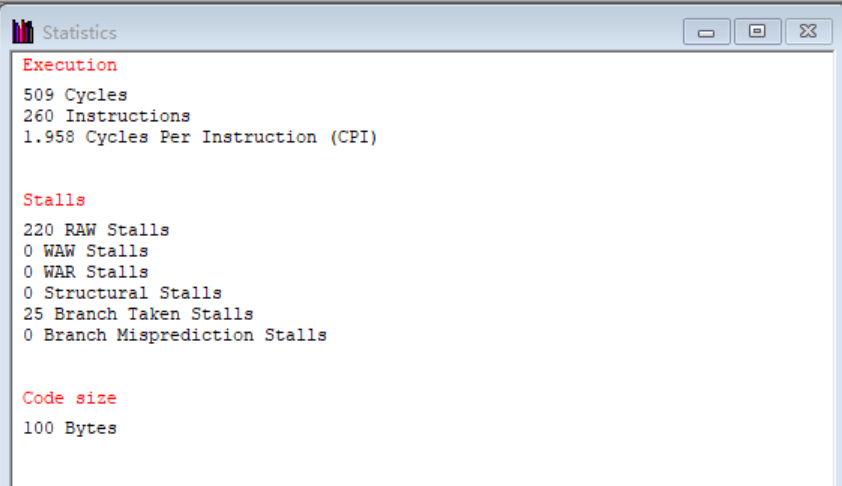
C[i][j] = A[i][j] + B[i][j];

根据上述的C代码，我们将其转换成MIPS语言，然后运行，并进行分析。

MIPS代码如下：



将这段代码加载到WinMIPS64中，运行结果如下（Statistic窗口结果）：



从Statistic窗口可以看出，本程序运行过程中总共产生了220处RAW的数据相关。接下来，我们一一列举出来这些数据相关产生的代码，并进行分析和优化。

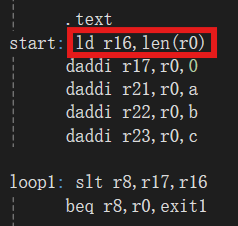
1. **调整指令序列**

在这一部分，我们进行优化的方式为调整指令序列:

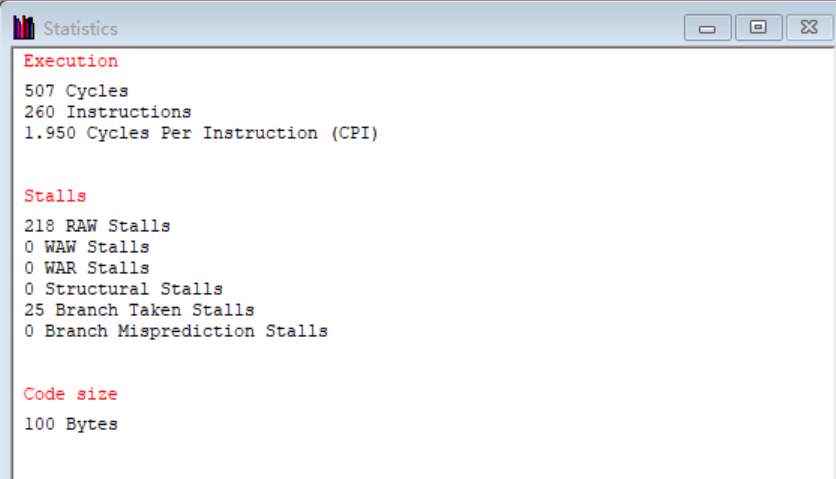
**第一个调整的部分：**



上诉的两条语句产生了两个RAW的阻塞，此处，我们可以通过将“ld r16，len（r0）”这条语句换到上面，即start：处，如下：

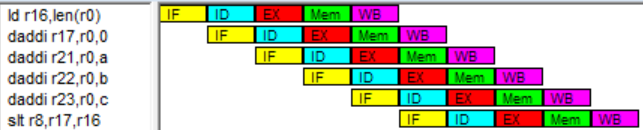


通过调整序列来规避这个数据相关，效果如下：



将此图与上面的结果图进行对比，我们可以发现，RAW阻塞的次数减少了两次。

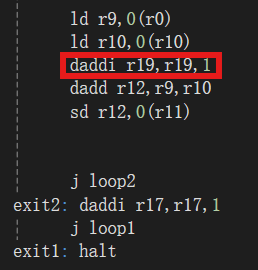
这样优化的目的是为了提前加载r16的值。在原始代码中，因为r16的值在loop1:中被立即使用，该指令需要等待数据写入寄存器r16后的下一个时钟周期才能进行读操作，产生了两个RAW的阻塞，不添加阻塞的话就会导致2b类冒险。而移到start:处可以提前加载r16的值，使后续的slt r8, r17, r16 语句在没有数据相关的情况下顺利执行，从而减少RAW阻塞，提高流水线效率。



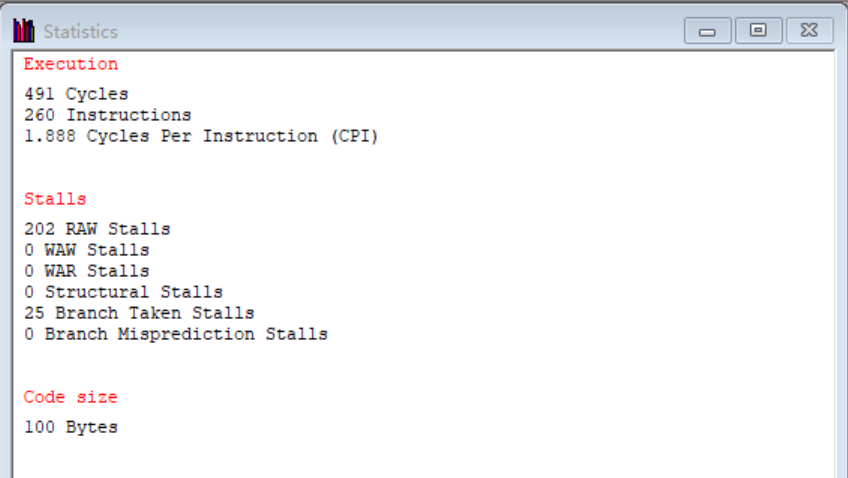
**第二个调整的部分：**



上述的语句产生了四个RAW数据相关，此处，我们也进行序列的调整，调整如下：



通过将“daddi r19，r19，1”这条语句上移，我们可以减少一次RAW的数据相关，运行后结果如下：



将结果图与之前的结果对照，很明显发现，RAW阻塞的次数减少了16次。效果还是比较明显。

鉴于此程序相对简单，可以通过调整指令序列来优化流水线相关的地方较少，但是上面的一些调整还是能明显的看出这种优化的有效性。

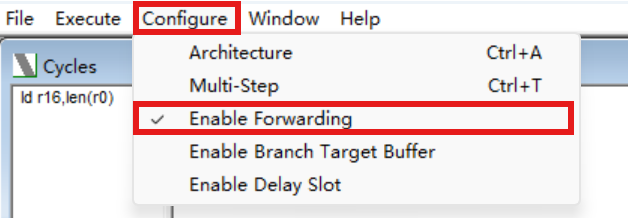
这样的优化方式能够尽量减少RAW阻塞，因为dadd r12, r9, r10指令需要等待寄存器r9、r10都回写后的下一个时钟周期读取这两个寄存器的值，产生两个RAW阻塞。同样也导致后续的指令受阻塞影响。而通过先进行没有数据相关的daddi r19, r19, 1指令，就能够代替一次RAW的数据相关，减少流水线中的停顿，提高指令吞吐量。



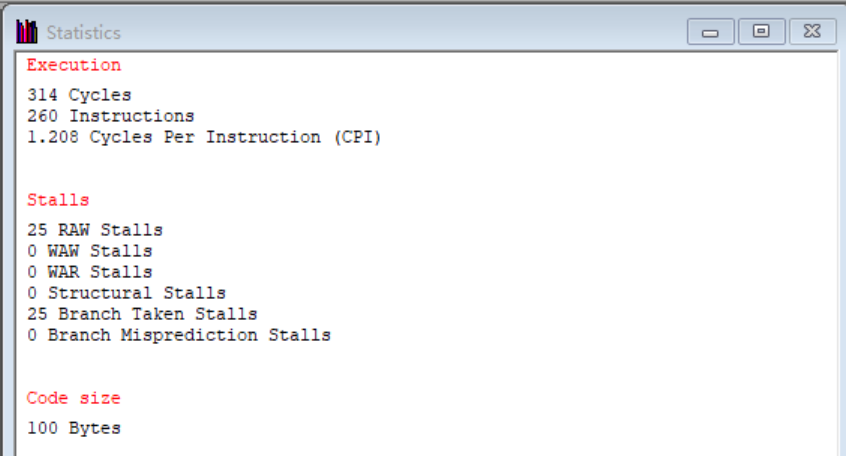
1. **Forwarding功能开启**

接下来，我们要展示Forwarding功能的优化效果。

首先，我们要知道如何开启Forwarding功能。方法如下：点开***configure***下拉窗口，给***Enable Forwarding***选项左侧点上勾。



开启了Forwarding功能之后，我们再运行，结果如下：



查看结果，可以看到RAW阻塞从202次变成了25次，效果十分显著。

Forwarding指的是旁路技术，用于减少流水线中数据相关。其通过将计算结果直接从执行阶段传递到后续需要该结果的指令，从而减少流水线的停顿，提高效率。例如第一次调整的部分，slt r8, r17, r16指令因为需要读取寄存器r16的值，所以在寄存器r16回写完成后还需阻塞一个时钟周期才能执行。借助旁路技术，就能够在回写寄存器r16的同时，传递回写的值给slt指令，减少了一次RAW阻塞。

**二、结构相关优化**

流水线中的结构相关，指的是流水线中多条指令在同一时钟周期内争用同一功能部件现象。即因硬件资源满足不了指令重叠执行的要求而发生的冲突。

在WinMIPS64中，我们可以在除法中观察到这种现象。要消除这种结构相关，我们可以采取调整指令位置的方法进行优化。在这个部分，我们首先给出几条C代码，然后将该代码翻译成MIPS代码（为了观察的方便，我们这里MIPS代码并不是逐一翻译，而是调整代码，使得其他部分数据相关已经优化，而两条除法指令连续出现），运行并查看结果。接着，调整代码序列，重新运行。观察优化效果。

下面是给出的C代码：

a = a / b

c = c / d

e = e + 1

f = f + 1

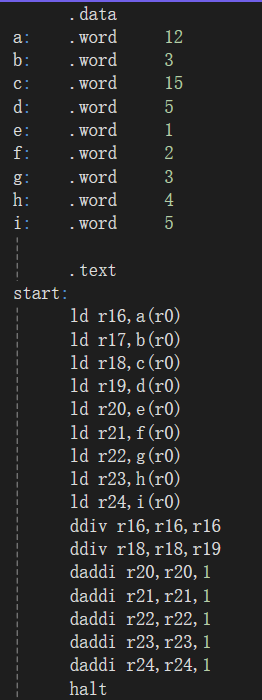
g = g + 1

h = h + 1

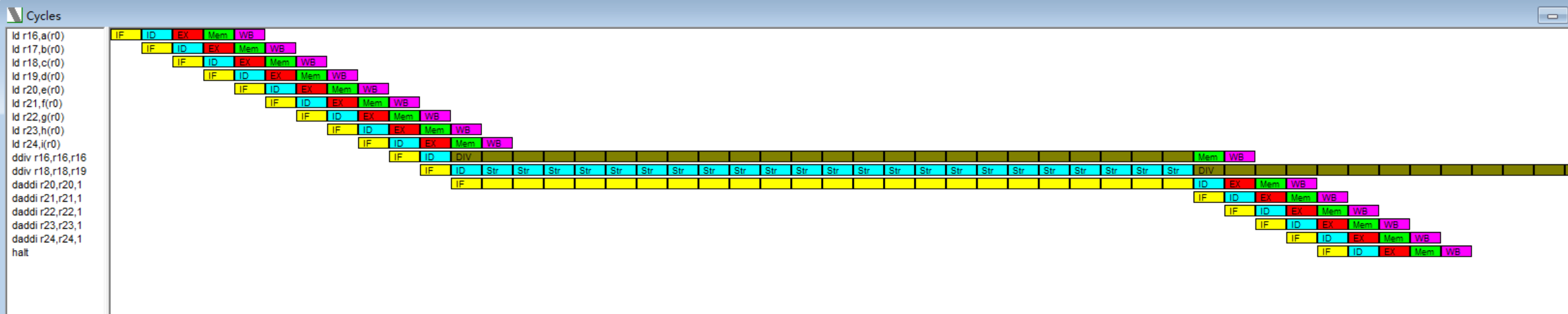
i = i + 1

j = j + 1

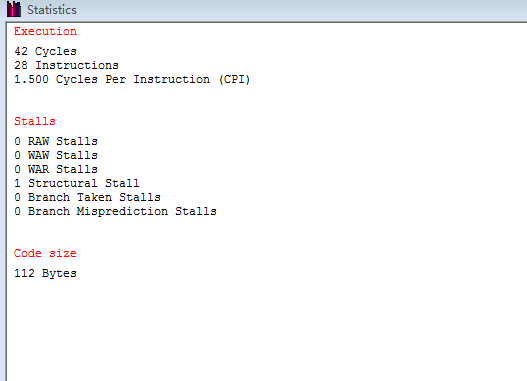
根据上述的C代码，我们给出数据相关优化的指令如下：



上面的指令运行，在***Cycle***窗口结果如下：



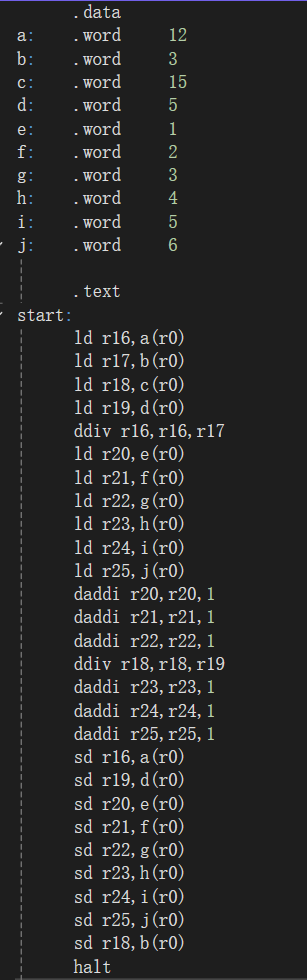
在***Statistics***窗口的结果如下：



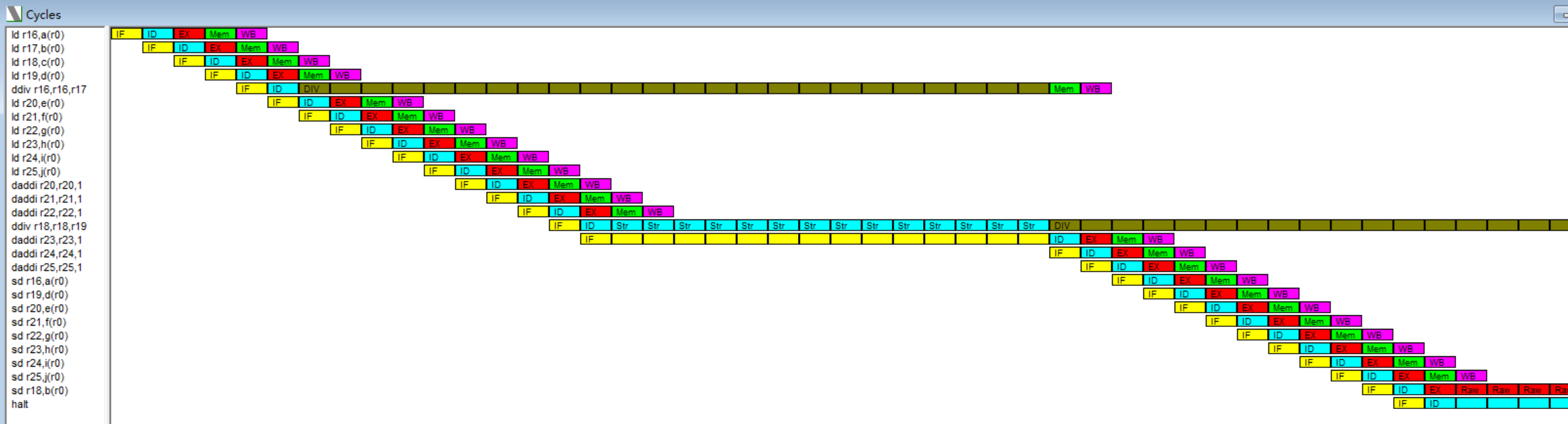
通过观察，我们可以发现，两个连续的除法产生了明显的结构相关，第二个除法为了等待上一个除法指令在执行阶段所占用的资源，阻塞了9个周期。

显然，这样的连续的除法所导致的结构相关极大的降低了流水线效率，为了消除结构相关，我们需要做的是调整指令序列，将其他无关的指令塞入两条连续的除法指令中。

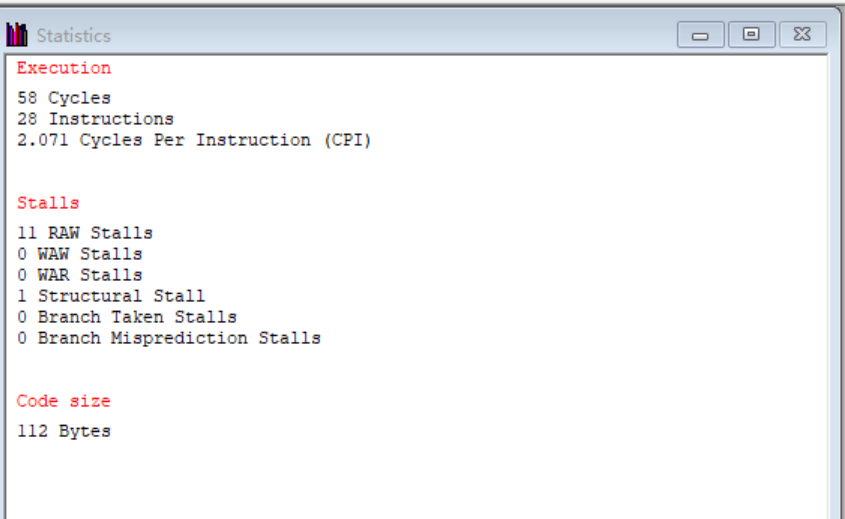
代码调整如下：



运行调整后的代码，在***Cycle***窗口结果如下：

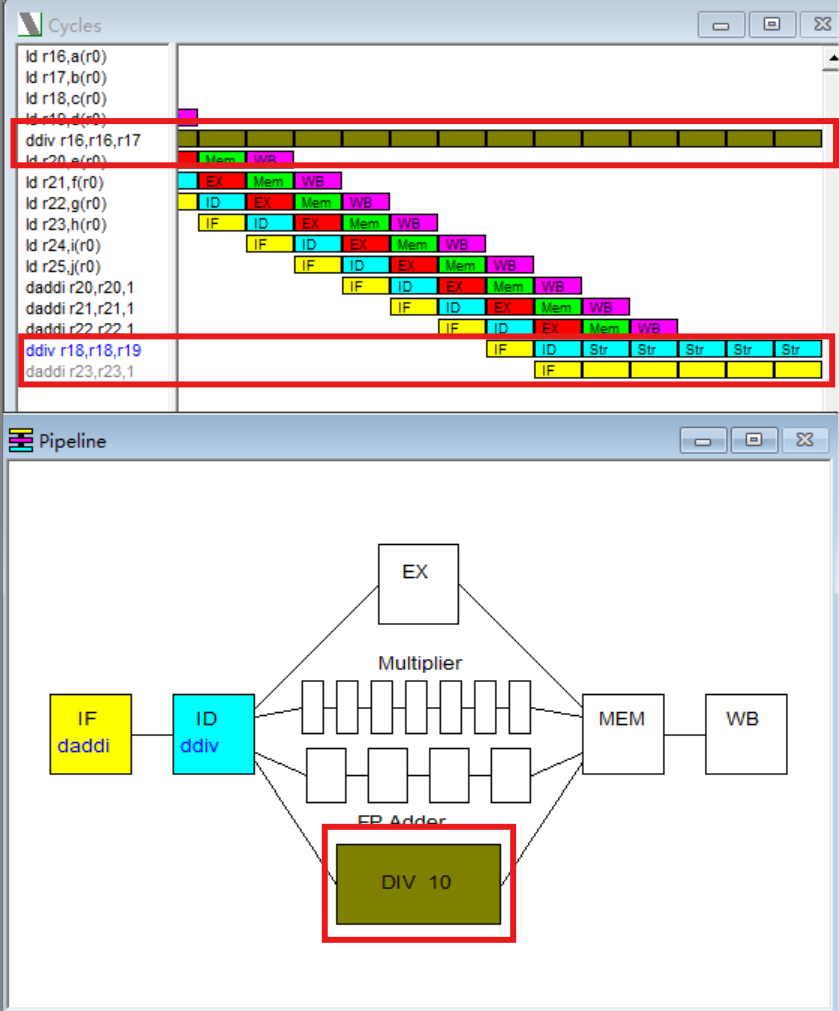


在***Statistics***窗口的结果如下：



通过对比，我们可以发现，优化后的代码在运行时减少了9个周期，优化效果显著。

在最初的代码中，有两条连续的ddiv指令都需要使用除法器，由于除法器在同一时钟周期内只能处理一条指令，第二条ddiv指令必须等待第一条完成，导致流水线停顿，阻塞了9个时钟周期。优化方法是让许多其他无关的ld以及add指令插入连续的ddiv指令之间，这样在除法器被占用时流水线先执行其他指令，尽可能地减少了争用资源引发的流水线停顿，提高了程序效率。



**五、实验总结与体会**

通过本次实验，我深入了解了MIPS五级流水线的工作原理和常见的流水线冒险类型。通过调整指令序列、启用Forwarding功能以及优化结构相关问题，改善了流水线停顿问题，显著提高了代码的执行效率。实验过程中，我不仅对MIPS指令有了更加深刻的理解，还学会了如何通过优化技术提高程序性能。

|  |
| --- |
| **指导教师批阅意见：**  **成绩评定：**  指导教师签字：  年 月 日 |
| 备注： |